ある拡張文法/オートマトンに関するコメント(計算アルゴリズムと計算量の基礎理論)

AUTHOR(S):

守屋, 悦朗

CITATION:

守屋, 悦朗. ある拡張文法/オートマトンに関するコメント(計算アルゴリズムと計算量の基礎理論). 数理解析研究所講究録 1989, 695: 21-26

ISSUE DATE:

1989-06

URL:

http://hdl.handle.net/2433/101409

数理解析研究所講究録
695巻 1989年 21-26

ある拡張文法/オートマトンに関するコメント

東京女子大学（文理）　守屋悦朗（Etsuro Moriya）

§0　はじめに

文脈自由文法（ＣＦＧ）の拡張として様々のものが知られている［7］中でも、インデックス文法［1］は最も自然な拡張と言えよう。ＣＦＧの持つ望ましい性質（例えば、導出木が描ける、最左導出と導出木とが1対1に対応する、多くの決定問題が帰納的に可解である、など）の多くを保存する唯一の拡張といってもよい。

さて、インデックス文法は、各非終端記号がプッシュダウンスタックを持っているようなＣＦＧと考えることが出来る。従って、自然の拡張として、非終端記号の持つ記憶機構を拡張することが考えられる。実際、既に西沢［5］がそのことをきわめて一般的に扱っているが、ここでは具体的な例として、1本ないし2本のスタック（またはキュー、またはカウンター）を持ったＣＦＧを考察する。また、それ等の文法をオートマトンで特徴付ける。このオートマトンは、［6］で導入された partitioning オートマトンのプッシュダウンオートマトン版である。

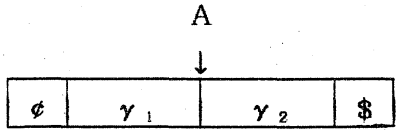
§1　スタックを持つＣＦＧ

はじめに、メモリを持つ文脈自由文法（ＣＦＧＭ）を一般的に定義する。　ＣＦＧＭ (context-free grammar with memory) は、$G=(N, \Sigma, \Gamma, P, S, \not\in, \$)$ なる5項組によって指定される。ここに、

(1) N は非終端記号の有限集合、
(2) Σ は終端記号の有限集合、
(3) Γ はメモリ記号の有限集合、
(4) $S\in N$ は文記号、
(5) $\not\in, \$\in N\cup\Sigma\cup\Gamma$ はメモリのエンドマーカであり、
(6) P は次のいずれかの形のプロダクションよりなる有限集合である：

①	$A\to\alpha$		$(A\in N;\ \alpha\in(N\cup\Sigma)^*)$
②	(i) $A\to Bf$	(ii) $A\to fB$	$(A,B\in N;\ f\in\Gamma)$
③	(i) $Af\to\alpha$	(ii) $fA\to\alpha$	$(A\in N;\ \alpha\in(N\cup\Sigma)^*)$
④	$Af\to Bg,\ Af\to gB,\ fA\to gB,\ fA\to Bg$		$(A,B\in N;\ f,g\in\Gamma)$

直感的には、ＣＦＧＭとは、次のように各非終端記号がメモリを持っているような文脈自由文法と考えることが出来る。文形式（$\phi S\$$ から導出される$(\phi \Gamma^* N \Gamma^* \$ \cup \Sigma)^*$の元）$\alpha \phi \gamma_1 A \gamma_2 \$ \beta$（ただし、$\alpha, \beta \in (\phi \Gamma^* N \Gamma^* \$ \cup \Sigma)^*$；$\gamma_1, \gamma_2 \in \Gamma^*$）において、各非終端記号 A はメモリ内容 $\gamma_1 \gamma_2$ を持ち、メモリポインタは γ_1 の最右文字（または、γ_2 の最左文字）を指していると考える。ϕ と$は、メモリの範囲を明示するためのメタシンボルであり、メモリの内容ではない。



G における導出を定義しよう。$(\phi \Gamma^* N \Gamma^* \$ \cup \Sigma)^*$ の上の２項関係⇒を次のように定義する。$\beta, \gamma \in (\phi \Gamma^* N \Gamma^* \$ \cup \Sigma)^*$； $\delta, \varepsilon \in \Gamma^*$； $A_1, \ldots, A_k \in N \cup \Sigma$ に対して、

１） $A \to A_1 \ldots A_k$ が①型プロダクションのとき、

$\beta \phi \delta A \varepsilon \$ \gamma \Rightarrow \beta \delta_1 A_1 \varepsilon_1 \ldots \delta_k A_k \varepsilon_k \gamma$

ここに、$A_1 \in N$ のとき $\delta_1 = \phi \delta$, $\varepsilon_1 = \varepsilon \$$； $A_1 \in \Sigma$ のとき $\delta_1 = \varepsilon_1 = \lambda$

すなわち、プロダクション左辺の非終端記号のメモリ内容はプロダクション右辺の各非終端記号に等しく継承される。

２） $A \to Bf$ (or $A \to fB$) が②型のプロダクションのとき、

$\beta \phi \delta A \varepsilon \$ \gamma \Rightarrow \beta \phi \delta Bf \varepsilon \$ \gamma$ (or $\beta \phi \delta A \varepsilon \$ \gamma \Rightarrow \beta \phi \delta fB \varepsilon \$ \gamma$)

これは、メモリへの書き込みプロダクションである。

３） $Af \to A_1 \ldots A_k$ (or $fA \to A_1 \ldots A_k$) が③型のプロダクションのとき、

$\beta \phi \delta Af \varepsilon \$ \gamma \Rightarrow \beta \delta_1 A_1 \varepsilon_1 \ldots \delta_k A_k \varepsilon_k \gamma$

(or $\beta \phi \delta fA \varepsilon \$ \gamma \Rightarrow \beta \delta_1 A_1 \varepsilon_1 \ldots \delta_k A_k \varepsilon_k \gamma$)

ここに、$A_1 \in N$ のとき $\delta_1 = \phi \delta$, $\varepsilon_1 = \varepsilon \$$； $A_1 \in \Sigma$ のとき $\delta_1 = \varepsilon_1 = \lambda$

これは、メモリ内容消去のためのプロダクションである。

４） $\zeta \to \theta$ が④型のプロダクションのとき、

$\beta \phi \delta \zeta \varepsilon \$ \gamma \Rightarrow \beta \phi \delta \theta \varepsilon \$ \gamma$

これは、メモリポインタ移動（同時に、メモリ内容も変え得る）のためのプロダクションである。

$\Rightarrow^*$を⇒の反射推移閉包とするとき、

$$L(G) = \{ w \in \Sigma^* \mid \phi S\$ \Rightarrow^* w \}$$

を、G の生成するＣＦＬＭという。

さて、ＣＦＧＭの持つメモリ構造はきわめて一般的なので、次のようにいくつかの制限を考える。

（ａ）④型のプロダクションを含まないとき、ＣＦＧ２Ｐ (context-free grammar with two pushdown stores)という。γ_1の部分がＡの左プッシュダウンストア（γ_1の最右文字がトップ）にあたり、γ_2の部分が右プッシュダウンストア（γ_2の最左文字がトップ）にあたる。④型のプロダクションがないので、メモリの書き換えはいつもγ_1とγ_2の境界のところで行なわれる。

（ｂ）ＣＦＧ２Ｐで、かつ、$\Gamma=\{¥, Z\}$、かつ、任意の文形式が $(\not\in ¥Z^*NZ^*¥\$ \cup \not\in NZ^*¥\$ \cup \not\in ¥Z^*N\$ \cup \not\in N\$ \cup \Sigma)^*$ の形をしているようなＣＦＧＭを ＣＦＧ２Ｃ (context-free grammar with two counters) という。¥は、プッシュダウンストアの底を表す。

（ｃ）以下では、②(ii)、③(ii)型のプロダクションを許さず、かつ、④型のプロダクションは次の形のものだけに制限する：

$$Af\to Bf,\quad Af\to fB,\quad fA\to fB,\quad fA\to Bf$$

すなわち、④型のプロダクションはメモリポインタの移動（スタックreadingに相当する）だけに使えメモリ内容の書き換えは出来ない。さらに、②(ii)型プロダクション A→Bf、③(ii)型プロダクション Af→B が適用出来るのは、次の形の文形式に限定する：

$$\beta\not\in A\delta\$\gamma \quad \text{または} \quad \beta\not\in Af\delta\$\gamma$$

$$(A,B\in N;\ f\in\Gamma;\ \delta\in\Gamma^*;\ \beta,\gamma\in(\not\in\Gamma^*N\Gamma^*\$\cup\Sigma)^*)$$

すなわち、どの非終端記号もプッシュダウンストアを１つだけ持ち（それは右プッシュダウンストアである）、②(i)型プロダクションによるメモリへの書き込み（プッシュダウンに相当）および、③(i)型プロダクションによるメモリの消去（ポップアップに相当）は右プッシュダウンストアのトップでのみ行なわれる。このようなＣＦＧＭを ＣＦＧＳ (context-free grammar with a stack) という。

（ｄ）④型のプロダクションを含まないＣＦＧＳをＣＦＧＰ (context-free grammar with a pushdown store) という。これは、インデックス文法に他ならない。

（ｅ）ＣＦＧＰで、かつ、$\Gamma=\{¥, Z\}$、かつ、任意の文形式が $(\not\in NZ^*¥\$ \cup \not\in N\$ \cup \Sigma)^*$ の形をしているようなものをＣＦＧＣ (context-free grammar with a counter)という。

これ等以外にも、ＣＦＧＳＣ (context-free grammar with a stack-counter [３])、ＣＦＧＱ (context-free grammar with a queue) 等も自然に定義出来るが省略する。Ｘ型の文法によって生成される言語のクラスを $\mathcal{L}_X$ で表すことにする。

定理１　$\mathcal{L}_{CFG2C}=\mathcal{L}_{CFG2P}=\mathcal{L}_{CFG2Q}=\mathcal{L}_{CFGM}$＝帰納的可算言語のクラス

証明は、任意のチューリングマシンは2-counter machineによってシミュレート出来るので、2-counter machine をＣＦＧ２Ｃでシミュレート出来ることをいえばよい。詳細は省略する。なお、この結果は一部［５］で既に示されている。

定理２　$\mathcal{L}_{CFG}\subseteq\mathcal{L}_{CFGC}\subseteq\mathcal{L}_{CFGP}\subseteq\mathcal{L}_{CFGS}\subset\mathcal{L}_{CSG}$

証明は省略するが、例えば

$\{ a^n b^n c^n \mid n \geqq 0 \} \in \mathcal{L}_{CFGC} - \mathcal{L}_{CFG}$、

$\{ ww \mid w \in \{a, b\}^* \} \in \mathcal{L}_{CFGP} - \mathcal{L}_{CFGC}$、

$\{ (w\#)^{|w|} \mid w \in \Sigma^* \} \in \mathcal{L}_{CFGS} - \mathcal{L}_{CFGP}$

である。なお、$\mathcal{L}_{CFGS} \subseteq \mathcal{L}_{CSG}$かどうかは不明であるが、ＣＦＧＳに対する emptimess problemはrecursively solvableであろうと予想しており、それが示せれば$\mathcal{L}_{CFGS} \subseteq \mathcal{L}_{CSG}$である。

§２　分割的オートマトン

分割的オートマトン(partitioning automaton)は、池川［６］によって、有限オートマトンとプッシュダウンオートマトンに対して初めて導入された概念である。ここではスタックオートマトンに対して分割的スタックオートマトンを定義し、それによってＣＦＧＳが特徴付けられることを示す。

ＰＳＡ (partitioning stack automaton) とは、$M = (Q, U, \Sigma, \Gamma, \delta, q_0, Z_0, F)$のことである。ここに、

１）Ｑは状態の有限集合、

２）$U \subset Q$はpartitioning stateの集合、

３）Σは入力記号の有限集合、

４）Γはスタック記号の有限集合、

５）$q_0 \in Q$は初期状態、

６）$Z_0 \in \Gamma$は初期スタック記号、

７）$F \subset Q$は受理状態の集合、

８）δは $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma$ から $\bigcup_n (Q \times \Gamma^*)^n \cup \bigcup_n (Q \times \{-1, 1\})^n$ の有限部分集合への関数（遷移関数）である。

$w \in \Sigma^*$ がＭに受理されるのは、次のような計算木Ｔが存在する場合である：

(i)　Ｔの各節にはＭの様相　($Q \times \Sigma^* \times \Gamma^* \{\vdash\} \Gamma^*$の元）がラベル付けされている。

(ii)　Ｔの根のラベルは初期様相　($q_0, w, \vdash Z_0$) である。

(iii) Ｔのどの葉のラベルも受理様相 ($F \times \{\lambda\} \times (\Gamma^* \{\vdash\} \Gamma^*$の元) である。

(iv)　πをラベル ($p, ax, Z_1 \ldots \vdash Z_1 \ldots Z_k$) を持つ、Ｔの内点とする。ただし、$a \in \Sigma \cup \{\lambda\}$、$x \in \Sigma^*$、各 $Z_j \in \Gamma$ とする。次の２通りの場合がある。

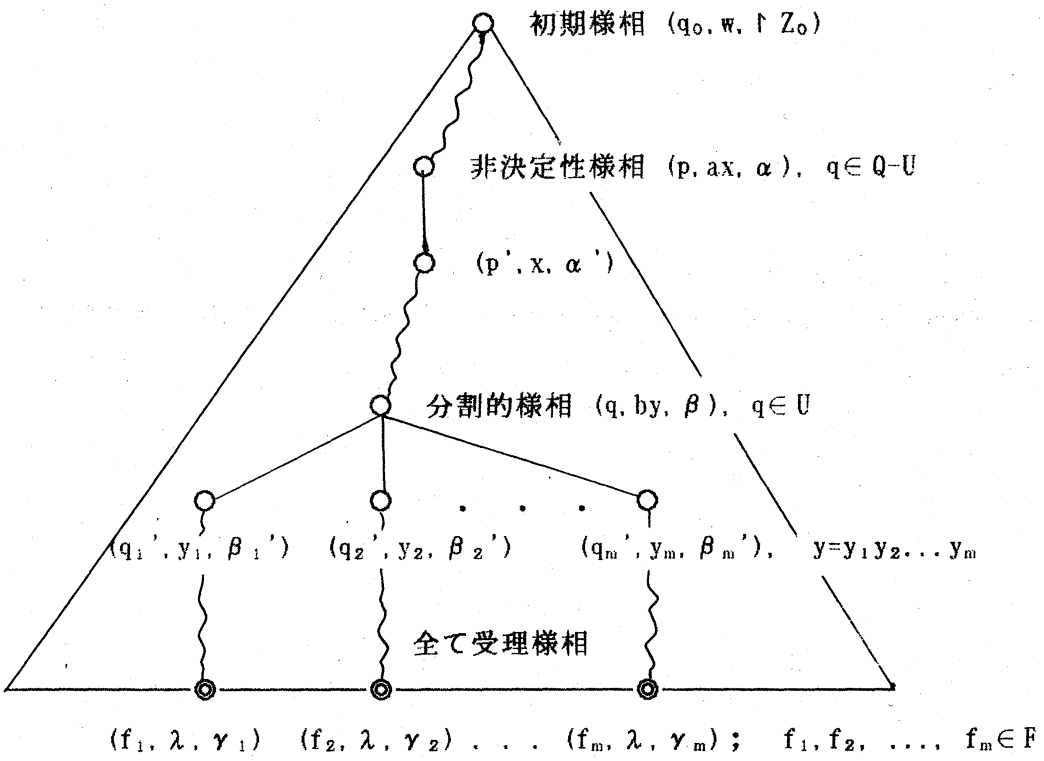
(iv-1) $p \in Q-U$ のとき非決定性モードという。πの子供は一人だけで、そのラベルは

(a: プッシュダウンモード)　($q, x, \vdash \gamma Z_2 \ldots Z_k$) である。ただし、i=1 かつ $\delta(p, a, Z_1)$ が $Q \times \Gamma^*$ の元 (q, γ) を含んでいるとする。

(b: スタック読みモード)　　$(q, x, Z_1 \ldots \vdash Z_{i+d} \ldots Z_k)$ である。ただし $\delta(p, a, Z_i)$ が $Q \times \{-1, 1\}$ の元 (q, d) を含んでいて、 $1 \leqq i+d \leqq k$ であるとする。

(iv-2) $p \in U$ のとき分割モードという。π の子供はm人で、それ等のラベルは左から順に $(q_1, x_1, \alpha_1), \ldots, (q_m, x_m, \alpha_m)$ である。ただし、$\delta(p, a, Z_i) = \{(q_1, r_1), \ldots, (q_m, r_m)\}$ で、かつ、各 α_j は (q_j, r_j) が $Q \times \Gamma^*$ の元であるか $Q \times \{-1, 1\}$ の元であるかに従って、(iv-1)(a)または(iv-1)(b)のように定義されたものであり、 $x = x_1 \ldots x_m$ でなければならない。

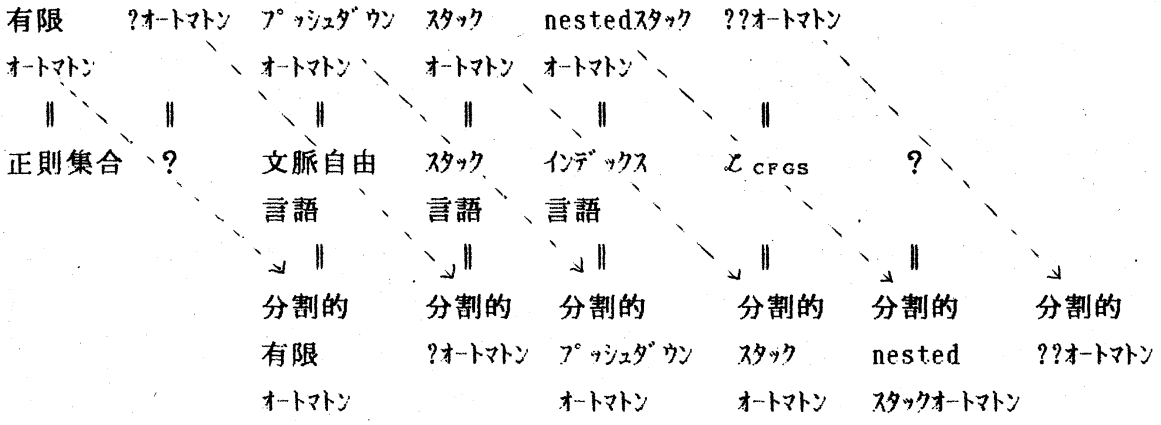
分割的オートマトンはalternatingオートマトンの１変形であると考えてよい。partitioning state (Uの元) はalternating stateに相当し、Q-U の元はexistential stateに対応する。オートマトンがexistential stateにいるときは、その様相から移り得る次の様相はδによって与えられる遷移可能な様相の中から非決定的に１つ選ばれる（iv-1: 非決定性モード）のに対し、partitioning stateを持つ様相からは、δによって与えられる遷移可能な様相の全てに計算が引き継がれ、以後計算はパラレルに進められるという点はalternatingオートマトンと同じであるが、partitioningな様相（計算木における親様相）からの遷移の際入力記号列のまだ読まれていない部分が分割されて子様相達に渡される点（iv-2: 分割モード）がpartitioningオートマトンたる名前の所以である。パラレルに進められた計算の全てが受理様相に至ったとき、入力記号列は受理される（iii）。



Mが受理する語の全体を L(M) で表し、ＰＳＡ言語と呼ぶ。上記は、スタックオートマトンに対してそのpartitioning版を定義したものであるが、同様にして、いろいろのオートマトンに対してそのpartitioning版が定義出来る。

定理３　ＰＳＡ言語のクラスは $\mathcal{L}_{CFGS}$ と一致する。

同様に、分割的counter machine言語のクラス＝$\mathcal{L}_{CFGC}$ が示される。池川［６］の結果（分割的有限オートマトンの受理する言語のクラス＝$\mathcal{L}_{CFG}$、分割的プッシュダウンオートマトンの受理する言語のクラス＝$\mathcal{L}_{CFGP}$(インデックス言語のクラス)）と合わせると、partitioningによってオートマトンの受理能力が１つシフトアップすることが分かる。



文献

1. A.V.Aho, Indexed grammars - an extension of context-free grammars, J.ACM 15 (1968), 647-671.
2. A.V.Aho, Nested stack automata, J.ACM 16 (1969), 383-406.
3. R.V.Book and S.Ginsburg, Multi-stack-counter languages, Math. Systems Theory 6 (1972), 37-48.
4. S.Ginsburg, S.A.Greibach and M.A.Harrison, One-way stack automata, J.ACM 14 (1967), 389-418.
5. 西沢輝泰、General indexed grammarとmonitored pushdown stack acceptor、日本数学会応用数学分科会、1979年4月.
6. M.Ikekawa, Modified one-way alternating pushdown automata and indexed languages, Trans. IECE Japan E69 (1986), 1213-1216.
7. A.Salomaa, "Formal Languages", Academic Press, 1973.